A PROBLEM OF FINDING AN ACCEPTABLE VARIANT IN GENERALIZED PROJECT NETWORKS

DAVID BLOKH, GREGORY GUTIN, AND ANDERS YEO

Received 20 January 2003 and in revised form 20 June 2003

A project network often has some activities or groups of activities which can be performed at different stages of the project. Then, the problem of finding an optimal/acceptable time or/and optimal/acceptable order of such an activity or a group of activities arises. Such a problem emerges, in particular, in house-building management when the beginnings of some activities may vary in time or/and order. We consider a mathematical formulation of the problem, show its computational complexity, and describe an algorithm for solving the problem.

1. Introduction

A project network may well has some activities which can be performed at different stages of the project. In such cases, we wish to compute an optimal (or, at least, acceptable) time or order of such an activity. Such problems appear, in particular, in house-building management when the beginnings of some activities may vary in time or/and order (see [1]), in scheduling of medical examinations in areas with difficult ecological situations [7, 9], and in building/city evacuation management [2, 10]. For other applications, see [3, 4, 6].

Consider a mathematical formulation of the problem. Let $G = (N,A)$ be a network, that is, finite acyclic digraph with unique source s. We will allow networks to have parallel arcs, that is, arcs with the same endvertices. The arc set A of G is partitioned into disjoint subsets D, A_i $(i = 1,2,\ldots,r)$: $A = \bigcup_{i=1}^{r} A_i \cup D$. A set A_i is called an *alternating* set. An arc from $\bigcup_{i=1}^{r} A_i$ is called an *alternating* arc. A *variant* $G' = (N',A')$ of G is a maximal subgraph of G satisfying the following conditions: s is a unique source of G', $|A' \cap A_i| = 1$ for all $i = 1,2,\ldots,r$. Note that because of the maximality of G', if distinct vertices x and y are in N' and an arc a between them is in D, then $a \in A'$.

Associate with each arc a of G its *time* $t(a)$ and *cost* $c(a)$. The *cost* of a variant G' is the sum of the costs of arcs from G' and the *time* of G' is the time of a critical path (see [8]) in G'. Given constants C and T, one wishes to find a variant (if any) whose cost is at most C and time is at most T (the *AV problem*). Any such variant is called *acceptable*.

Journal of Applied Mathematics and Decision Sciences 2005:2 (2005) 75–81
DOI: 10.1155/JAMDS.2005.75

We prove that the problem of finding an acceptable variant is NP-complete (even if G has a very simple structure). Moreover, we show that the problem of verifying whether G has a variant (any variant) is also NP-complete. Therefore, the existence of a polynomial algorithm for solving the AV problem seems to be impossible. We describe an algorithm which generates all variants in a clever way in order to avoid construction of some non-variants. Having a variant, we check whether it is acceptable. The algorithm might be useful for moderate inputs.

The problem of finding an optimal variant in very special networks was first stated by Eisner [4]. Variants in general networks were considered in [6], yet every alternating set in [6] is formed by arcs with common initial vertex.

2. Complexity

Proposition 2.1. *The AV problem is NP-complete.*

Proof. We transform the well-known NP-complete problem, KNAPSACK [5, page 247], to this problem. Let a set $U = \{1,2,\dots,r\}$, and given integer size $c_i \geq 0$ and value $v_i \geq 0$, for every $i \in U$, as well as positive integers B and K, constitute an arbitrary instance of KNAPSACK. In KNAPSACK, we wish to find a subset $W \subseteq U$ such that $\sum_{i\in W} c_i \leq B$ and $\sum_{i\in W} v_i \geq K$. Let $\bar{W} = U - W$ and $V = \sum_{i=1}^{n} v_i$. Then, KNAPSACK is equivalent to the problem of determining a set $W \subseteq U$ such that $\sum_{i\in W} c_i \leq B$ and $\sum_{j\in \bar{W}} v_j \leq V - K$.

On the other hand, the last problem is clearly equivalent to the following special case of the AV problem: let $G = (N,A)$ be the network with $N = \{x_1,x_2,\dots,x_{r+1}\}$, $A = \bigcup_{i=1}^{r} A_i$, where $A_i = \{a_i,a_i'\}$, a_i and a_i' are parallel arcs with initial vertex x_i and terminal vertex x_{i+1}. The cost of an arc a_i (a_i', resp.) is c_i (0, resp.) for all $i = 1,\dots,r$. The time of an arc a_i (a_i', resp.) is 0 (v_i, resp.). Set $C = B$, $T = V - K$. □

Now we consider the problem of checking whether a given network G has a variant (the EV problem). We first restrict ourselves to a slight extension of networks considered in the proof of Proposition 2.1, namely, to "multipath" networks with vertex set $N = \{x_1,x_2,\dots,x_{r+1}\}$ and arc set $A = \bigcup_{i=1}^{r} A_i$, where every arc in A_j has the form x_jx_{j+1} (A_j may contain several arcs). Then the EV problem is polynomial time solvable due to its simple transformation into the perfect matching (in bipartite graphs) problem. However, in general, the EV problem is NP-complete.

Theorem 2.2. *The EV problem is NP-complete.*

Proof. We transform the well-known problem 3-SAT [5, page 46] to the EV problem. Let $U = \{u_1,\dots,u_k\}$ be a set of variables, let $C = \{c_1,\dots,c_m\}$ be a set of clauses such that every c_i has three literals, and let v_{il} be the lth literal in the clause c_i. We will construct a network G which has a variant if and only if C is satisfiable.

We construct network G as follows. We start with the vertices $s,x_1,\dots,x_{2k}$ and arcs $sx_1,\dots,sx_{2k}$. For every $i = 1,\dots,k$ and every $j = 1,\dots,m$, if the variable u_i belongs to the clause c_j, we add the arc $x_{2i-1}y_{ij}$, where y_{ij} is a new vertex. Analogously, for every $i = 1,\dots,k$ and every $j = 1,\dots,m$, if the negation of u_i belongs to the clause c_j, we add the arc $x_{2i}z_{ij}$, where z_{ij} is a new vertex. The arcs sx_{2i-1},sx_{2i} constitute the set A_i ($i = 1,\dots,k$). All arcs of the forms $x_{2i-1}y_{ij}$ and $x_{2i}z_{ij}$ constitute the set A_{k+j} for $j = 1,\dots,m$.

We will prove that G has a variant if and only if C is satisfiable.

Suppose first that C is satisfiable and consider a truth assignment α for U that satisfies all the clauses in C. Let v_{j,l_j}, $j = 1,2,\ldots,m$, be true under α. Construct a variant G' (starting from empty G') as follows. For every $j = 1,2,\ldots,m$, if v_{j,l_j} is a variable u_i, then the arcs sx_{2i-1} and $x_{2i-1}y_{ij}$ (along with their vertices) are added to G', otherwise, that is, v_{j,l_j} is the negation of a variable u_i, the arcs sx_{2i} and $x_{2i}z_{ij}$ are added to G' (we do not append arcs which are already in G'). Clearly, G' contains exactly one arc from each of the sets $A_{k+1},\ldots,A_{k+m}$. Moreover, it has at most one arc from each of the sets $A_1,\ldots,A_k$ since a variable and its negation cannot be both true under α. However, G' may contain no arcs from some of the sets $A_1,\ldots,A_k$. After appending representatives of such sets to G' we clearly obtain a variant of G.

The above arguments can be "reversed" in order to prove that if G contains a variant, then C is satisfiable. □

3. Algorithm

In this section, we describe an algorithm, which finds all variants of a network. The algorithm is illustrated by an example in the full version of the paper in www.cs.rhul.ac.uk/home/gutin/. Let $G = (N,A)$ be a network, as described in the introduction. Set $A_{r+1} = D$ and let $(A_1,A_2,\ldots,A_r)$ be some ordering of the alternating sets in G. Furthermore, let $A_i = (a_1^{(i)},a_2^{(i)},\ldots,a_{|A_i|}^{(i)})$ be some ordering of the arcs in A_i $(i = 1,2,\ldots,r)$. Define the vertices $u_k^{(i)}$ and $v_k^{(i)}$, such that $a_k^{(i)} = u_k^{(i)}v_k^{(i)}$ for all $i = 1,2,\ldots,r$ and $k = 1,2,\ldots,|A_i|$. A *feasible r-tuple*, $\vec{\alpha} = (\alpha_1,\alpha_2,\ldots,\alpha_r)$ is an ordered set of integers, such that $1 \le \alpha_i \le |A_i|$ for all $i = 1,2,\ldots,r$. If $\vec{\alpha}$ is a feasible r-tuple then let $(\vec{\alpha})_j$ (or just α_j) denote the jth entry in the tuple $\vec{\alpha}$. If $\vec{\alpha}$ and $\vec{\beta}$ are feasible r-tuples, and there exists an integer $j \ge 1$, such that $\alpha_i = \beta_i$ for all $i = 1,2,\ldots,j-1$ and $\alpha_j < \beta_j$, then we say that $\vec{\alpha} \prec \vec{\beta}$ and that $\omega(\vec{\alpha},\vec{\beta}) = \omega(\vec{\beta},\vec{\alpha}) = j$. Finally, if $\vec{\alpha}$ is a feasible r-tuple, then $A^*(\vec{\alpha}) = \{a_{\alpha_1}^{(1)},a_{\alpha_2}^{(2)},\ldots,a_{\alpha_r}^{(r)}\}$.

We now show the following lemma.

Lemma 3.1. *Given a feasible r-tuple $\vec{\alpha}$, we can, in $O(|N|^2)$ time, decide if there is a variant $G' = (N',A')$, with $A^*(\vec{\alpha}) \subseteq A'$.*

Furthermore if such a variant exists, then it is unique.

Proof. Define a network G^*, a vertex set B, and an arc set C as follows:

$$\begin{aligned} G^* &= \left(V(G),A^*\left(\vec{\alpha}\right) \cup A_{r+1}\right), \\ B &= \{b : \text{there is a path from } s \text{ to } b \text{ in } G^*\}, \\ C &= A^*\left(\vec{\alpha}\right) \cup \{uv : uv \in A_{r+1} \text{ and } \{u,v\} \subseteq B\}. \end{aligned} \tag{3.1}$$

It is easy to check that there is a variant (containing the arcs from C) in G^* if and only if $\{u_{\alpha_i}^{(i)},v_{\alpha_i}^{(i)}\} \subseteq B$, for all $i = 1,2,\ldots,r$. Furthermore using normal breadth-first search, B can be found in $O(|N|^2)$ time, which implies that C can be found in $O(|N|^2)$ time. This proves the first part of the lemma.

The second part follows immediately since if there is a variant in G^*, then it must be precisely the one containing all the arcs in C. □

The trivial algorithm now consists of generating all feasible r-tuples, and for each one testing if there is a corresponding variant, using Lemma 3.1. This implies an algorithm which is quite simple and runs in $O(|N|^2 \Pi_{i=1}^{r} |A_i|)$ time. However, in many cases, we can do better than this. Before we describe our algorithm we need a few definitions and lemmas.

For a network $G = (N, A)$ and set B, $G\langle B\rangle$ denotes the subgraph of G *induced* by B.

Definition 3.2. Let $\vec{\alpha}$ be a feasible r-tuple and $G' = G\langle A^*(\vec{\alpha}) \bigcup_{k=j}^{r+1} A_k\rangle$. Define $m(\vec{\alpha}, v)$ for all vertices $v \in N$, as well as $p(\vec{\alpha})$ and $q(\vec{\alpha})$ in the following manner:

$$m(\vec{\alpha}, v) = \max\{j \le r+1 : \exists \text{ a path from } s \text{ to } v \text{ in } G'\}, \tag{3.2}$$

$$p(\vec{\alpha}) = \min\left\{\max\left\{m\left(\vec{\alpha}, u_k^{(i)}\right) : 1 \le k \le |A_i|\right\} : 1 \le i \le r\right\}, \tag{3.3}$$

$$q(\vec{\alpha}) = \min\left\{\max\left\{m\left(\vec{\alpha}, u_{\alpha_i}^{(i)}\right), i\right\} : 1 \le i \le r\right\}. \tag{3.4}$$

Observe from Definition 3.2 that a variant including the arcs $\{a_{\alpha_1}^{(1)}, a_{\alpha_2}^{(2)}, \ldots, a_{\alpha_l}^{(l)}\}$ $(0 \le l \le r)$ cannot include any vertex v, with $m(\vec{\alpha}, v) \le l$. The definition of $p(\vec{\alpha})$ implies that $p(\vec{\alpha})$ is the largest value, such that for every alternating set, A_i, there is an arc $u_k^{(i)} v_k^{(i)} \in A_i$ with $m(\vec{\alpha}, u_k^{(i)}) \ge p(\vec{\alpha})$. Similarly, $q(\vec{\alpha})$ is the smallest number, such that there is an $i \in \{1, 2, \ldots, r\}$ with $m(\vec{\alpha}, a_{\alpha_i}^{(i)}) \le q(\vec{\alpha})$ and $i \le q(\vec{\alpha})$. The meaning of $m(\vec{\alpha}, v)$, $p(\vec{\alpha})$, and $q(\vec{\alpha})$, should hopefully become clear after Lemma 3.3.

Lemma 3.3. *Let $\vec{\alpha}$ and $\vec{\beta}$ be feasible r-tuples, with $\omega(\vec{\alpha}, \vec{\beta}) = l$. Then the following hold.*

(i) *Any variant including the arcs $A^*(\vec{\beta})$ cannot include any vertex v, with $m(\vec{\alpha}, v) < l$.*

(ii) *If $l > p(\vec{\alpha})$, then there is no variant including the arcs $A^*(\vec{\beta})$.*

(iii) *If $l > q(\vec{\alpha})$, then there is no variant including the arcs $A^*(\vec{\beta})$.*

(iv) *Given $\vec{\alpha}$ we can compute $m(\vec{\alpha}, v)$ for all $v \in N$, $p(\vec{\alpha})$ and $q(\vec{\alpha})$ in $O(|N|^2)$ time.*

Proof. (i) Any variant including the arcs $A^*(\vec{\beta})$ also includes the arcs $\{a_{\alpha_1}^{(1)}, a_{\alpha_2}^{(2)}, \ldots, a_{\alpha_{l-1}}^{(l-1)}\}$. By the remarks after Definition 3.2, there is no path from s to v, in $G\langle A^*(\vec{\beta}) \cup A_{r+1}\rangle$, for any vertex v with $m(\vec{\alpha}, v) \le l - 1 < l$.

(ii) If $l > p(\vec{\alpha})$, then there exists an alternating set A_i, with $\max\{m(\vec{\alpha}, u_k^{(i)}) : k = 1, 2, \ldots, |A_i|\} = p(\vec{\alpha}) < l$. Therefore $m(\vec{\alpha}, u_k^{(i)}) < l$, for all arcs $u_k^{(i)} v_k^{(i)} \in A_i$. By (i), no arc in A_i lies in any variant including $A^*(\vec{\beta})$, so there is no variant including $A^*(\vec{\beta})$.

(iii) If $l > q(\vec{\alpha})$, then there is an $i \in \{1, 2, \ldots, r\}$, such that $\max\{m(\vec{\alpha}, u_{\alpha_i}^{(i)}), i\} = q(\vec{\alpha}) < l$. This implies that $m(\vec{\alpha}, u_{\alpha_i}^{(i)}) < l$ and $i < l$. It follows that $a_{\alpha_i}^{(i)}$ does not lie in any variant including $A^*(\vec{\beta})$, so there is no such variant.

(iv) Let $G_r = G\langle A^*(\vec{\alpha}) \cup A_{r+1}\rangle$. Using breadth-first search on G_r, we find all the vertices V_r which can be reached from s in G_r. We then contract all the vertices in V_r (note $s \in V_r$) into the vertex s. Then we add all the arcs from A_r to obtain the graph G_{r-1}. Using

breadth-first search on G_{r-1}, we find all the vertices V_{r-1} which can be reached from s in G_{r-1}. Continuing this process, we obtain the sets $V_r, V_{r-1}, V_{r-2}, \ldots, V_1$. Clearly if a vertex v is not in $V_r \cup V_{r-1} \cup \cdots \cup V_{k+1}$, but $v \in V_k$, then $m(\vec{\alpha}, v) = k$. Furthermore, the time used in total is $O(\sum_{k=1}^{r} |V_i|^2) = O(|N|^2)$.

To compute $p(\vec{\alpha})$, we first compute $\text{temp}(i) = \max\{m(\vec{\alpha}, u_k^{(i)}) : k = 1,2,\ldots,|A_i|\}$, for each $i = 1,2,\ldots,r$, in $O(\sum_{i=1}^{r} |A_i|)$ time. Then we can compute $p(\vec{\alpha})$ in $O(r)$ time using the values $\text{temp}(i)$, which implies that $p(\vec{\alpha})$ can be computed in $O(|N|^2)$ time.

We can compute $q(\vec{\alpha})$ in $O(|N|^2)$ time as we can get $\max\{m(\vec{\alpha}, u_k^{(i)}), i\}$ in constant time for each $i = 1,2,\ldots,r$. This completes the proof of the lemma. □

We now present our algorithm.

Step 1. Let $\vec{\alpha}_1 = (1,1,\ldots,1)$ be a feasible r-tuple (it has r ones). Let $c = 1$.

Step 2. Use Lemma 3.1 to decide whether there is a variant including the arcs $A^*(\vec{\alpha}_c)$. If there is a variant, then print it.

Step 3. Compute $m(\vec{\alpha}_c, v)$ for all $v \in N$, and compute $p(\vec{\alpha}_c)$ and $q(\vec{\alpha}_c)$ (see Definition 3.2 and Lemma 3.3). Let $\vec{\beta} = \vec{\alpha}_c$ and $k = \min(p(\vec{\alpha}_c), q(\vec{\alpha}_c), r)$.

Step 4. Set $\beta_k = \beta_k + 1$. While $m(\vec{\alpha}_c, u_{\beta_k}^{(k)}) < k$ and $\beta_k \le |A_k|$ increase the value of β_k by one. Afterwards if $\beta_k > |A_k|$, then decrease the value of k by one and stop if $k = 0$, or return to the start of Step 4 otherwise.

Step 5. For $j = k+1, k+2, \ldots, r$, let β_j be the smallest value such that $m(\vec{\alpha}_c, u_{\beta_j}^{(j)}) \ge k$.

Step 6. Let $c = c+1$, $\vec{\alpha}_c = \vec{\beta}$ and return to Step 2.

Theorem 3.4. *The above algorithm finds all possible variants in $O(C|N|^2)$ time, where C is the value of c when the algorithm terminates.*

Proof. We first show that, in Step 5, there is always a value β_j, such that $m(\vec{\alpha}_c, u_{\beta_j}^{(j)}) \ge k$, for all $j = k+1, k+2, \ldots, r$. Assume that this is false, then $\max\{m(\vec{\alpha}_c, u_{\beta_j}^{(j)}) : \beta_j = 1,2,\ldots,|A_j|\} < k$, which implies that $k > p(\vec{\alpha}_c)$. This is however a contradiction against the definition of k, and the fact that the value of k never increases in Step 4. As $\vec{\alpha}_1 \prec \vec{\alpha}_2 \prec \cdots \prec \vec{\alpha}_C$, this implies that the algorithm terminates.

We will show that if there is a feasible r-tuple $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_r)$, and an integer c, such that $\vec{\alpha}_c \prec \vec{\gamma} \prec \vec{\alpha}_{c+1}$, then there is no variant including the arcs $A^*(\vec{\gamma})$. The case when $\vec{\alpha}_C \prec \vec{\gamma}$ (C is the maximum value of c) will then be analogous. This will imply the theorem since we have used Lemma 3.1 to check if there is a variant including a feasible r-tuple, for all feasible r-tuples which may be included in a variant.

Let $x = \omega(\vec{\alpha}_c, \vec{\gamma})$, $z = \omega(\vec{\alpha}_{c+1}, \vec{\gamma})$. Consider two cases.

Case 1. $x > \min(p(\vec{\alpha}_c), q(\vec{\alpha}_c), r)$.

We have either $x > p(\vec{\alpha}_c)$ or $x > q(\vec{\alpha}_c)$, which by Lemma 3.3 implies that there is no variant including the arcs $A^*(\vec{\gamma})$.

Case 2. $x \le \min(p(\vec{\alpha}_c), q(\vec{\alpha}_c), r)$.

Note that every new r-tuple $\vec{\alpha}_{c+1}$ equals current $\vec{\beta}$ in Step 6 . Note that by Step 4 performed last time before Step 6 , $k = \omega(\vec{\alpha}_c, \vec{\alpha}_{c+1})$. We have $z > k$ as $\vec{\alpha}_c < \vec{\gamma} < \vec{\alpha}_{c+1}$.

We now prove that $m(\vec{\alpha}_c, u_{\gamma_z}^{(z)}) < k$. Then, by Lemma 3.3, there is no variant including the arcs of $A^*(\vec{\gamma})$. Assume that $m(\vec{\alpha}_c, u_{\gamma_z}^{(z)}) \geq k$ and consider the following three subcases.
Subcase 2.1 ($x > z$). By the assumption of Case 2, $x \leq \min(p(\vec{\alpha}_c), q(\vec{\alpha}_c), r)$. Clearly, $\gamma_x > (\vec{\alpha}_c)_x$. Therefore, in Step 4, the algorithm would have chosen $k \geq x$. Thus, $k \geq x > z$ which implies $k > z$. However, $(\vec{\alpha}_{c+1})_z > \gamma_z = (\vec{\alpha}_c)_z$ (as $x > z$), which means that $z \geq k$; a contradiction.
Subcase 2.2 ($x = z$). Hence, $(\vec{\alpha}_c)_z < \gamma_z < (\vec{\alpha}_{c+1})_z$. Thus, Step 4 would have terminated with $\beta_z = \gamma_z$ rather than $(\vec{\alpha}_{c+1})_z$; a contradiction.
Subcase 2.3 ($x < z$). We have $(\vec{\alpha}_c)_x < \gamma_x = (\vec{\alpha}_{c+1})_x$ (as $x < z$). It follows that $k \leq x < z$. Thus, in Step 5, β_z would have been smaller than or equal to γ_z rather than $(\vec{\alpha}_{c+1})_z$; a contradiction.

We have now shown that if there is a variant in G, then our algorithm must find it. It is easy to show that the complexity of our algorithm is $O(C|N|^2)$ (using Lemma 3.3). □

We note that C denotes a number of distinct feasible r-tuples, which implies that $C \leq \Pi_{i=1}^{r} |A_i|$. Therefore our algorithm will never take more time asymptotically, than the trivial algorithm. In some cases our algorithm will however be considerably faster than the trivial algorithm. The speed of our algorithm also depends on the ordering of the alternating sets $A_1, A_2, \ldots, A_r$. We will not use great length to describe an optimal ordering of the alternating sets, however it is always a good idea to let the alternating sets which can reach many other alternating edges have small indices.

Using the proof of Theorem 2.2, we can transform every instance of the 3-SAT problem into a network. We call the networks obtained in this way *3-SAT networks*. We will now see how much time we may need to find a variant in a 3-SAT network. Let V denote the number of variables in an instance of the 3-SAT problem and let C be the number of clauses. Let $N = V + C$. Clearly the trivial algorithm takes $O(N^2 2^V 3^C)$ time. For comparison we prove the following theorem, which shows that our algorithm is considerably faster than the trivial algorithm in some cases.

THEOREM 3.5. *Let G be a 3-SAT network. If we order the alternating sets, such that the sets corresponding to variables have smallest indices, then our algorithm takes $O(N^2 2^V)$ time to find a variant in G.*

Proof. Let the alternating sets $A_1, A_2, \ldots, A_V$ correspond to variables and the alternating sets $A_{V+1}, A_{V+2}, \ldots, A_{V+C}$ correspond to clauses. Let $r = V + C$ and let $\vec{\alpha}_c = (\alpha_1, \alpha_2, \ldots, \alpha_r)$ be a feasible r-tuple at the cth iteration of our algorithm. Let the ith variable be true if and only if $\alpha_i = 1$. If some clause is false, then we note that all arcs uv in the corresponding alternating set have $m(\vec{\alpha}_c, u) \leq V$. Therefore our algorithm will have $p(\vec{\alpha}_c) \leq V$, which implies that no two tuples in $\{\vec{\alpha}_1, \vec{\alpha}_2, \ldots, \vec{\alpha}_C\}$ can have the same values in the first V places. Therefore $C \leq 2^V$, which implies the theorem. □

It can be shown that the value of q in our algorithm does not speed up the algorithm, except possibly in the first iteration. After the first iteration we will always have $q(\vec{\alpha}_c) \geq \min(p(\vec{\alpha}_c), r)$.

References

[1] H. M. Ahuja, *Construction Performance Control by Networks*, John Wiley & Sons, New York, 1976.

[2] L. G. Chalmet, R. L. Francis, and P. B. Saunders, *Network models for building evacuation*, Management Sci. **28** (1982), no. 1, 86–105.

[3] E. A. Dinic, *The fastest algorithm for the PERT problem with AND-and OR-nodes*, Proc. Integer Prog. and Combinatorial Opt. Conference, University of Waterloo Press, Waterloo, 1990, pp. 185–187.

[4] H. Eisner, *A generalized network approach to the planning and scheduling of a research project*, Oper. Res. **10** (1962), no. 1, 115–125.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, California, 1979.

[6] N. A. J. Hastings and J. M. C. Mello, *Decision Networks*, John Wiley & Sons, Chichester, 1978.

[7] V. K. Ivanov, A. I. Gorsky, A. F. Tsyb, M. A. Maksyutov, and E. M. Rastopchin, *Dynamics of thyroid cancer incidence in Russia following the Chernobyl accident*, J. Radiol. Prot. **15** (1990), 305–318.

[8] E. Minieka, *Optimization Algorithms for Networks and Graphs*, Industrial Engineering, vol. 1, Marcel Dekker, New York, 1978.

[9] K. B. Moysich, R. J. Menezes, and A. M. Michalek, *Chernobyl-related ionising radiation exposure and cancer risk: an epidemiological review*, Lancet Oncol. **3** (2002), no. 5, 269–279.

[10] Z. Sinuany-Stern and E. Stern, *Simulating the evacuation a small city: the effects of traffic factors*, Socio-Econ. Plann. Sci. **27** (1993), no. 2, 99–108.

David Blokh: Department of Physics, Bar-Ilan University, Ramat Gan 52900, Israel
E-mail address: davidb@bgumail.bgu.ac.il

Gregory Gutin: Department of Computer Science, Royal Holloway University of London, Egham, Surrey TW20 0EX, UK
E-mail address: gutin@cs.rhul.ac.uk

Anders Yeo: Department of Computer Science, Royal Holloway University of London, Egham, Surrey TW20 0EX, UK
E-mail address: anders@cs.rhul.ac.uk